**一、线程管理(java.lang.Thread)**

**1创建线程**

Thread匿名类/子类实现

Thread t = new Thread() {

public void run() { //code... }

}

Runnable实例创建Thread

Runnable r = new Runnable() {

public void run() { //code... }

}

Thread t = new Thread(new);

**2线程信息**

线程对象

static int activeCount() 返回活跃线程数

static Thread currentThread() 返回当前线程

long getId() 返回线程的标识符

String getName() 返回线程的名称(可在构造中指定)

setName(String) 设置线程名

堆栈与加载器

static dumpStack() 打印当前线程堆栈信息

static Map getAllStackTraces() 返回存活线程的堆栈信息(映射集)

StackTraceElement[] getStackTrace() 返回线程的堆栈信息

ClassLoader getContextClassLoader() 返回线程的类加载器

setContextClassLoader(ClassLoader) 设置线程的类加载器

**二、线程调度(java.lang.Thread)**

**1线程中断**

中断信息

static boolean interrupted() 测试当前线程是否已中断

boolean isInterrupted() 判断线程是否中断(检查被标记状态)

中断方法

**2线程跃迁**

跃迁信息

int getPriority() 返回线程的优先级(1~10默认5)

setPriority(int) 设置线程优先级

Thread.State getState() 返回线程的状态

boolean isAlive() 判断线程处于活动状态

跃迁方法

sleep(long) 睡眠线程

start() 启动线程

yield() 释放资源(让其他同优先级的执行)

**3线程同步**

同步执行

join(<long>) 线程被同步执行(直到线程退出或超时)

同步锁：多线程同一时间，只有一个线程能够进入临界区

·对象锁：表达式synchronized(obj){}，以obj区分锁(临界区)

·实例方法锁：public synchronized void xx(){}，以方法的实例区分锁(临界区)

·静态方法锁：public static synchronized yy(){}，以静态方法的类区分锁(临界区)

信号控制：以对象锁为例(实例/静态方法锁用实例/类替换)

obj.wait() 释放锁

obj.notify() 给出一个信号，唤醒一个wait锁的线程

obj.notifyAll() 给出信号，唤醒所有wait锁的线程

同步队列：见LinkedBlockingQueue(先进先出)、LinkedBlockingDeque(双端队列)

**三、线程池**

**1控制并发(线程worker)**

系统提供：见Semaphore，可指定并发线程数目

手动实现

public class HelloWorld {

static int pool\_size = 3;

static int current = 0;

public static void main(String args[]) throws Exception {

for(int i=0; i<100; i++) {

new Thread(() -> { //java8 lambda表达式

try {

// 运行满负荷时，等待获取信号

synchronized (HelloWorld.class) {

if(current == pool\_size) HelloWorld.class.wait();

current++;

}

// 线程要执行的部分

System.out.println("I'm doing ..."); Thread.sleep(1000);

# 释放一个信号，唤醒等待线程

synchronized (HelloWorld.class) {

current--;

HelloWorld.class.notify();

}

} catch (Exception e) {}

}).start();

}

}

}

**2线程数量**

基于定长安全队列：

·新建线程：向队列推入一个元素

·队列满时，推入操作将被阻塞，即最大线程数的控制

**3系统提供线程池**

线程池构造器：Executors类，用于构造线程池对象

newCachedThreadPool() 无限大线程池(ThreadPoolExecutor)

newFixedThreadPool(int) 指定大小线程池(ThreadPoolExecutor)

newScheduledThreadPool(int) 计划任务线池(ScheduledThreadPoolExecutor)

线程池接口：

·Executor：提供execute(Runnable)执行任务

·ExecutorService：继承Executor，提供Future submit(Callable)执行任务

·ScheduledExecutorService：继承ExecutorService，提供schedule(...)执行计划任务

线程池类

·ThreadPoolExecutor：实现了Executor、ExecutorService，可自定义大小和队列

·ScheduledThreadPoolExecutor：实现了ScheduledExecutorService